CS257 Linear and Convex Optimization

Homework 6

Due: November 2, 2020

October 26, 2020

For this assignment, you should submit a pdf file as well as your source code (.py files). The pdf file should include all necessary figures, the outputs of your Python code, and your answers to the questions. Do NOT write your answers in any of the .py files. Please TYPE your answers. You may use Latex, Word, or any other software you prefer.

In this assignment, you will implement gradient descent with constant step size. First complete the function `gd_const_ss` in `gd.py`.

1. Consider the following quadratic program,

$$\min_{\boldsymbol{x}\in\mathbb{R}^2} f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} \tag{1}$$

where

$$\boldsymbol{Q} = \begin{pmatrix} 1 & 0 \\ 0 & \gamma \end{pmatrix}.$$

with $\gamma > 0$. In terms of coordinates,

$$f(x_1, x_2) = \frac{1}{2}x_1^2 + \frac{\gamma}{2}x_2^2.$$

We know the optimal solution is $\boldsymbol{x}^* = \boldsymbol{0}$ and $f^* = f(\boldsymbol{x}^*) = 0$.

(a). What is the smallest L such that f is L-smooth?

(b). Suppose $\gamma = 10$. Complete **p1.py** and run the gradient descent algorithm you implemented with step sizes 0.22, 0.1, 0.01, 0.001. (Note: for step size 0.22, you may want to limit the maximum number of iterations to a small number, say 10.) Does it converge in each case? When it does converge, how many iterations does it take? In each case, plot the 2D trajectory of the sequence $\boldsymbol{x}_k$ and the function values $f(\boldsymbol{x}_k)$. You can use the function provided in `utils.py` or write your own code. Note that $f(\boldsymbol{x}_k)$ is also the approximation error since $f(\boldsymbol{x}^*) = 0$.

(c). For $\gamma = 1, 0.1, 0.01, 0.001$, run gradient descent with step size 1. How many iterations do you need in each case? How does the number of iterations change as γ decreases?

2. Consider the least squares problem in Problem 3(a) of Homework 5, i.e.

$$\min_{\boldsymbol{w}} \|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|_2^2$$

where

$$\boldsymbol{X} = \begin{pmatrix} 2 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}, \quad \boldsymbol{y} = \begin{pmatrix} 3 \\ 2 \\ 2 \end{pmatrix}.$$

Use your implementation of gradient descent to solve this least squares problem. You may follow provided codes of problem 1. Use any stepsize you find appropriate. Compare your solution with the solution you found in HW5, and the solution you find by solving the normal equation using `np.linalg.solve`.

3. ***Logistic regression*** is a statistical model that uses a logistic function to analyze the relationship between a binary dependent variable and one or more dependent variables. You are required to build a logistic-regression classifier for a toy dataset. Each sample in the dataset includes a 2-dimension vector that represents the feature of the sample and a binary label that identifies the type of the sample. The model takes the feature as input and predicts the type, and it can be expressed as:

$$\sigma_{\boldsymbol{\theta}}(\boldsymbol{x}) = \frac{1}{1 + e^{-\boldsymbol{\theta}^T \boldsymbol{x}}}$$

where $\boldsymbol{\theta}$ represents the parameters of the model and $\boldsymbol{x}$ is the input. Here we add a bias term to the model by having one more dimension for both $\boldsymbol{\theta}$ and $\boldsymbol{x}$ with a 1 appended to $\boldsymbol{x}$ (see the provided code), so feature vector of each data sample actually has 3 dimensions. If we take $\sigma_{\boldsymbol{\theta}}(\boldsymbol{x})$ as the probability that the sample belongs to type 1, then $1 - \sigma_{\boldsymbol{\theta}}(\boldsymbol{x})$ is the probability the sample belongs to type 0. The predicted label is determined by the one with the greater value, i.e. if $\sigma_{\boldsymbol{\theta}}(\boldsymbol{x}) > 1 - \sigma_{\boldsymbol{\theta}}(\boldsymbol{x})$ then the predicted type is 0.

The objective function of this problem is given below:

$$f(\boldsymbol{\theta}) = -\sum_{i=1}^{N} [y^i \log \sigma_{\boldsymbol{\theta}}(\boldsymbol{x}^i) + (1 - y^i) \log(1 - \sigma_{\boldsymbol{\theta}}(\boldsymbol{x}^i))],$$

where N is the total number of samples in the dataset, $\boldsymbol{x}^i$ and y^i are the feature and the label of the i^{th} sample respectively. We want to minimize this objective function:

$$\min_{\boldsymbol{\theta}} f(\boldsymbol{\theta}).$$

(a). Prove that the objective function is convex. Hint: Use the convexity of known functions and the convexity-perserving operations.

(b). Prove that the gradient of the objective function is

$$\sum_{i=1}^{N} (\sigma_{\boldsymbol{\theta}}(\boldsymbol{x}^i) - y^i) \boldsymbol{x}^i.$$

Hint: $g(x) = \frac{1}{1+e^{-x}}, \nabla g(x) = g(x)(1 - g(x))$.

(c). The toy dataset is given in **p3.py**. Optimize your model (using **gradient descent**) and test it on the dataset. Follow the guidance in **p3.py** and complete it. Use a **screenshot** to display the accuracy (the ratio of samples whose labels are predicted correctly by your model).

Hint: Use matrix operations to calculate the function values and gradients. Some numpy functions for matrix operations that might be useful are listed below. Refer to online documentation for their usage.

numpy.matmul(), *numpy.multiply()*, *numpy.exp()*, *numpy.reshape()*.